Project Background

Aslak and I were brought on to a project with the task of refactoring a legacy codebase. A complex application with approximately 30 thousands lines of code, written over a 2 year timeframe by many programmers. The code worked as intended and was actually a success compared to all previous efforts to accomplish the same sort of functionality. The team is *somewhat* agile and uses CruiseControl for automated builds.

The Goals

We had some clear directives from the client:

1. Disentangle the GUI portion of this codebase (AWT stuff) from the domain layer
2. Rewrite the GUI as a remote client
3. Expose the domain layer as a clean and reusable public API

At first glance, the code didn't look too badly written and Clover was reporting over 60% coverage from approximately 1100 Junit tests. Hmm... Diving into the test suite usually seems like a good starting point to figure out where to start.

Problematic Unit Tests

To my dismay, there were some serious issues with the way that the unit tests were written. I should say *so-called* u nit tests, cause there were some major problems to address:

- First of all, the test suite was taking about 10-15 minutes to execute.
- As a result, nobody in the team was running the suite on a regular basis except cruise control.
- Suite only passing in forked JVM mode, because of excessive use of statics requiring a pristine runtime. Executing the suit in-process reveals almost all clobber each other in non-deterministic ways.
- Unable to run suite in IntelliJ, because of the aformentioned issue with in-process execution, and because of hundreds of dependencies on test files (xml, db, properties, etc.) to be in the right place on the filesystem.
- About 20 singletons (oh cursed pattern!) and a lot of static utility classes
- Hundreds of mock, dummy, and faux subclasses. Tons of duplication and very confusing.

The majority of those tests weren't actually unit tests at all. Sure, they all extended JUnit TestCase, but most of them did some sort of file I/O, many included calls to Thread.sleep(), and during their execution there would be about 20-30 AWT windows popping up and flashing stuff in my face. As a matter of fact, they were *system tests* masquera ding as unit test.